Meshing Genus-1 Point Clouds Using Discrete One-Forms

Meshing Genus-1 Point Clouds using Discrete One-Forms

Geetika Tewari Craig Gotsman Steven J. Gortler

Computer Science Dept.
Harvard University

Abstract

We present an algorithm to mesh point clouds sampled from a closed manifold surface of genus 1. The method relies on a doubly-periodic global parameterization of the point cloud to the plane, so no segmentation of the point cloud is required. Based on some recent techniques for parameterizing higher genus meshes, when some mild conditions on the sampling density are satisfied, the algorithm generates a closed toroidal manifold which interpolates the input and is geometrically similar to the sampled surface.

Keywords: mesh generation, surface reconstruction, reverse engineering, point cloud, torus, parameterization.

1. Introduction

Point cloud meshing, sometimes called *surface reconstruction* or *reverse engineering*, is a procedure that takes a set of points sampled from a 3D surface, and reconstructs a triangle mesh that closely approximates the surface from which it was sampled, both in terms of its geometry and its topology. This problem is an important component in producing digital 3D models by 3D scanning.

The main problem with meshing point clouds is that the problem is essentially ill-posed. There is no one definition of what the true solution should be, beyond the general requirement that the mesh should look as similar as possible to the surface from which the points were sampled (which is usually not even known). In the few cases where an attempt has been made to define this, the solution is still far from unique.

2. Related Work

With the advent of high-precision 3D scanners, the last decade has seen a wealth of methods developed to mesh point clouds. Rather than survey them all exhaustively here (which would require many pages), we mention briefly just a few of the more well-known algorithms, and refer the interested reader to the excellent survey by Dey [10] for more details on these and other algorithms.

The first algorithm for meshing point clouds was described by Boissonnat in 1984 [4], but the problem received little attention until the work of Hoppe et al. [12] in 1992. These papers established two trends in surface reconstruction: *Delaunay-based* methods, where the surface is approximated by a subcomplex of a Delaunay complex (e.g. [1,2,5,9,23]), and *volumetric* methods, where the surface is approximated as the zero-set of a scalar 3D function (e.g. [8,6,25]).

Another distinction between various meshing algorithms relates to the basic objective of the algorithm. Some algorithms simply want to “connect the dots” to form a surface which *interpolates* the samples. Others do not require that the surface pass through the samples, rather *approximate* them in some sense. This is appropriate if the point cloud is noisy.

One of the main challenges to the interpolating meshing algorithms is to form a manifold (triangle) mesh whose vertices are the point cloud, having the correct topology and geometry close to that of sampled surface (the so-called “underlying manifold”). Once a manifold mesh is formed, it may be improved by a variety of post-processing techniques which improve the connectivity structure and possibly also the geometry of the mesh (by smoothing the points) while preserving manifoldness and the topology. In this way the effect of noise may also be reduced if it is present in the input, despite initially having interpolated the input.

A family of meshing algorithms which interpolates a point cloud was introduced by Floater and Reimers [14] for surfaces with disk topology, and then extended by Zwicker and Gotsman [31] for spherical surfaces. These rely on global parameterization of the point cloud, proceeding roughly as follows: 1) Construct a *k*-nearest neighbor graph (KNNG) on the point cloud. 2) Use the KNNG to parameterize the point cloud to a

natural parameter domain. 3) Triangulate the parameterized points in the parameter domain using some reasonable triangulation method (e.g. Delaunay). 4) Adopt the resulting mesh connectivity to triangulate the input 3D point cloud. For point clouds sampled from a manifold surface having the topology of a disk (i.e. possessing a boundary), Floater and Reimers parameterized to the plane, using the method of barycentric coordinates, which involves solving a set of linear equations. The equations essentially position each vertex at some convex combination of its neighbors' positions. This provides some degree of proximity in parameter space between vertices close to each other in the point cloud. Then, triangulating the points in parameter space using a triangulation routine which favors small compact triangles, will, in turn, induce a set of small 3D triangles on the sample set. This tends to produce a good approximation of the underlying manifold.

Realizing that a planar parameterization is not the most natural for a spherical point cloud, Zwicker and Gotsman [31] parameterized that type of input to the unit sphere, using an extension of the barycentric coordinates method, as described by Gotsman et al [17]. This involves the solution of a set of non-linear equations. For a toroidal point cloud, we will see later that it is possible to return to the plane, albeit in a non-trivial manner, and the algorithm that we present here is a member of the family of meshing techniques based on global parameterization.

3. Our Contribution

We extend recent results of Gu and Yau [20] (see also Gortler et al [16]) on the parameterization of closed manifold meshes of genus 1 to parameterize a point cloud sampled from such a surface. The parameterization is planar and doubly-periodic with no seams, in a sense to be made precise later, thus a planar triangulation of it may be used to mesh the point cloud in a seamless manner.

Our method does not necessarily produce results significantly better than other interpolating reconstruction algorithms. It does have, however, three main advantages: 1) It introduces some new mathematical tools to the surface meshing problem. 2) It is very simple to implement. 3) Under very mild conditions on the sampling density, it guarantees a closed manifold output with the correct topology. This mesh is geometrically very close to the sampled surface, and can be improved in an independent post-processing step.

4. The Algorithm

In order to apply the basic method of Floater and Reimers [14] to a toroidal point cloud, the first challenge is to identify a suitable parameter domain. Seeing that Zwicker and Gotsman [31] parameterized a spherical point cloud to the 3D unit sphere, the first thing that comes to mind is some sort of generic 3D torus shape. However, working on a 3D torus is quite difficult. An alternative is to partition the point cloud into segments, which can then be parameterized, as separate patches, to the plane. This is what Horman and Reimers [21] originally proposed for a spherical point cloud. However, such a partition would introduce seams and boundary artifacts into the result. Luckily, there is a way to parameterize a closed toroidal manifold mesh to the plane in a seamless way. Although this is a *global* method, in the sense that it involves computing information about the entire parameterization simultaneously, the information is ultimately applied *locally* to small pieces of the point cloud. Nonetheless, it is the global nature of the parameterization that guarantees the consistency between the pieces. This follows from results of Gu and Yau [20] and Gortler et al. [16], which we briefly describe in Section 4.1. We then provide the details of the different steps of the algorithm in the following subsections.

4.1 Harmonic one-forms on manifold meshes

Most parameterization methods for meshes with simple topologies pose a system of equations for the coordinates of the mesh vertices in some parameter space (plane or sphere). These coordinates are values attached to the vertices of the mesh. As we will see later, this is not directly applicable to toroidal meshes, and the key to parameterizing a toroidal mesh to the plane is to solve instead a system of equations for values attached to the *edges* of the graph. These values represent the *difference* in parameter values between the two vertices incident on that edge. Values defined on the edges of a graph are also known as a discrete *one-form*, in analogy to the continuous one-forms used in differential geometry [11,25].

Standard parameterization methods attempt to parameterize a manifold mesh to the plane. Typically they aim to flatten the mesh faces, such that the resulting faces in the plane are disjoint (do not overlap), and the distortion of their shapes relative to the 3D faces is minimal. The interested reader is referred to the survey by Floater and Hormann [15] for a comprehensive survey of different parameterization methods. Tutte [30] first showed how to parameterize a manifold mesh with disk topology by constraining the mesh boundary vertices to a convex shape, and solving a linear system of

equations for each of the (*x* and *y*) coordinate values of the interior vertices. The linear system implies that each interior vertex should be positioned in the plane at the centroid of its neighbor's positions. The physical analogy to this is a spring system of zero rest lengths, which relax to the desired rest position. Assume a planar graph *G* with *B* boundary vertices, *V* interior vertices, *E* edges and *F* faces. Tutte proved that if *G* is 3-connected, then the resulting faces in the plane form a non-degenerate *embedding*, namely, have positive area and are disjoint. This embedding also minimizes the sum of the squares of the edge lengths among all drawings of G in the plane with the same boundary conditions. Floater [13] later showed that this construction will work even if the edges are weighted, such that ultimately each interior vertex is positioned at x_i, which is some arbitrary *convex combination* of the positions of its neighbors:

$$\forall i \in \{1,..,V\} \quad x_i = \sum_{j \in N(i)} w_{ij} x_j \qquad \sum_{j \in N(i)} w_{ij} = 1, \ \forall j \ w_{ij} \geq 0$$

$N(i)$ is the set of neighbors to vertex *i*. Simple algebra shows that this is equivalent to:

$$\forall i \in \{1,..,V\} \quad 0 = \sum_{j \in N(i)} w_{ij} (x_i - x_j) \tag{1}$$

which implies that the vector *x*, consisting of the (*x* or *y*) coordinates of the interior *vertices*, is the solution to a discrete Laplace equation with (convex) boundary conditions, hence the discrete equivalent of a *harmonic* function.

Now let us look at (1) in a slightly different way: Define new *half-edge* variables $\Delta x_{ij} = x_i - x_j$. By definition $\Delta x_{ij} = -\Delta x_{ji}$. Eq. (1) is now equivalent to the following set of *co-closedness* equations for the new set of variables:

$$\forall v \in \{1,..,V\} \quad 0 = \sum_{e \in \delta v} w_e \Delta x_e \tag{2}$$

where *v* is a vertex, *e* a half-edge between two vertices, and δv the set of half-edges emanating from *v*. So, in principle, instead of solving the set of *V* linear equations in *V* unknowns (1) for *x*, we may solve the set of *V* linear equations in approximately *E* unknowns (2) for Δx (subject to equivalent boundary conditions). However, since *E*>*V*, this set (2) is seriously underdetermined, and this becomes apparent when we realize that we are missing some other constraints on the vector Δx. These are the constraints that will force them to fit together to faces, namely that all faces are *closed*:

$$\forall f \in \{1,..,F\} \quad 0 = \sum_{e \in \partial f} \Delta x_e \tag{3}$$

Where *f* is a face and ∂f is the set of half-edges bounding *f*. These are the *closedness* equations. Thus the combined set of equations (2) and (3) is now equivalent to the original set (1). Note that $\Delta x_{ij} = -\Delta x_{ji}$, so care must be exercised when treating the directions of edges (and, in fact, we need use explicit unknowns only for one half-edge per edge).

Now assuming we have solved (2)+(3) with the appropriate boundary conditions for the *one-form* Δx - how do we reconstruct the planar coordinates of the mesh vertices – the vector *x* ? One way to do this is by designating an arbitrary vertex in the mesh as the origin and *integrate* the one-form to every other vertex along some path of edges. This integrated value will be well-defined (unique) if the value of the integral is path-independent. This is guaranteed precisely because of (3). The value is then the vertex coordinate. Note that this procedure introduces a translational degree of freedom due to the choice of the origin.

In itself, this transformation of the equations for the vertex-based positions to equations for the edge-based one-forms does not seem to add anything when attempting to parameterize a disk-like mesh to the plane. Quite the opposite, it just makes things more complicated by increasing the size of the linear system from *V*×*V* to *E*×*E*. However, the power of this transformation emerges when we consider a closed manifold mesh with genus 1. First of all, there are no boundary conditions (so *B*=0 and all vertices are interior vertices). One-forms whose faces are all closed (Eq. (2)), and all vertices co-closed (Eq. (3)) with respect to some set of weights are called *harmonic*. The number of unknowns is *E*, and the number of equations is *V*+*F*, which, according to Euler's formula, is exactly *E*. This seems to imply that there are no harmonic one-forms but the trivial zero solution, but closer inspection reveals that if the weights in (2) are symmetric ($w_{ij} = w_{ji}$), the rank of the *V* co-closedness equations (2) is just *V*-1, and the rank of the closedness equations (3) is just *F*-1. This is because the co-closedness of all vertices but one implies the closedness of that last vertex, and the closedness of all faces but one implies the closedness of the last face. So the dimension of the linear space of harmonic one-forms for toroidal meshes is 2. A basis for this nullspace can be computed, and each basis vector used as the one-form which is later integrated to generate *x* and *y* coordinates respectively, when needed. This essentially is the parameterization method of Gu and Yau [20] applied to the toroidal case.

Note that the closedness equations imply that the sum of a harmonic one-form along any *boundary* of a set of faces of the mesh also vanishes. However, the same sum along a loop of edges which merely circles a "handle" of the mesh (but does not bound any set of faces) will not necessarily vanish.

A key consequence of this theory becomes apparent when, given two (linearly) independent harmonic one-forms, Δx and Δy, defined on the edges of a closed manifold toroidal mesh, we consider two submeshes having disk topology with an intersection which also has disk topology. In this case, the parameterization-by-integration procedure described above, when applied to Δx and Δy on each of the submeshes, results in two planar parameterizations which *coincide*, up to translation, on the vertices common to both subsets. This is a key feature of the parameterization, and guarantees that "pieces" of the mesh may be parameterized locally and independently, yet they will all fit together seamlessly at the global level, as illustrated in Figure 1. Furthermore, Gortler et al. [16] have shown that this parameterization results in a "Tutte-like" embedding of the submeshes in the plane, namely that the faces are all non-degenerate and disjoint, analogously to Tutte's theorem for the disk.

4.2 One-forms on arbitrary graphs

While the harmonic one-form theory described in the previous section is elegant and well-understood for manifold meshes, where the vertex and face structure is well-defined, it requires some modifications before it can be applied to parameterization of point clouds.

The first step in parameterizing a point cloud is to connect the points together into a graph. This is typically done using a *k*-nearest neighbor graph (KNNG). The result will generally not be planar or represent a manifold in any way (or even close to it). Indeed, an oriented manifold graph on the point cloud is precisely what we would like to generate in the output !

The main problem in applying the one-form theory developed in Section 4.1 to an arbitrary graph is the absence of well-defined faces to use for the closedness equations (3). However, it turns out that it is possible to do something similar when the *cycles* of the graph are used instead of the faces. Namely, we must force *all* cycles in the graph corresponding to boundary loops in the underlying surface to be closed – the integral of the one-form along that cycle should vanish.

There are an exponential number of cycles in a graph. Fortunately, in order that all these cycles be closed, it suffices to solve closedness equations for a *basis* of the cycles. By basis, we mean a minimal set of cycles that span all other cycles. In order that the term "span" be meaningful, we have to define a vector space. Assume an (arbitrary but fixed) orientation for each edge in the graph. Given a cycle of half-edges (e_1, e_2 ,.., e_m) of length m, express it as the vector $\Delta c \in \{-1,0,+1\}^E$, where

$$\Delta c_e = \begin{cases} 0 & e \notin \{e_1,..,e_m\} \\ +1 & \exists k \in \{1,..,m\}\ s.t.\ e = e_k, orient(e) = orient(e_k) \\ -1 & \exists k \in \{1,..,m\}\ s.t.\ e = e_k, orient(e) \neq orient(e_k) \end{cases}$$

It is well known [3] that *all* cycle vectors of a graph are members of a linear subspace of R^E of dimension $D=E-V+1$. Thus by computing a basis of this subspace $\{b_1$,.., $b_D\}$ and imposing the linear equation $\Delta x^T b=0$ for every such basis vector will force every cycle to be closed. It turns out, quite conveniently, that bases for this subspace may be formed from cycle vectors themselves (namely $b_i \in \{-1,0,+1\}^E$).

Assume that there exists an underlying genus-1 manifold such that the edges of the KNNG may be "drawn" on it as geodesic arcs. Then some of the cycles of the KNNG will correspond to boundaries (of surface regions) on the manifold, and some will not. The latter will "loop around the handles of the surface". We call

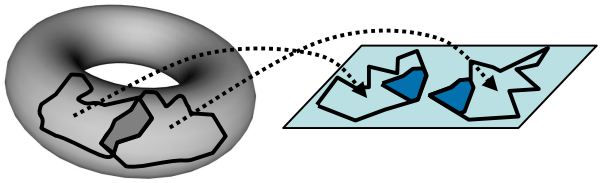

Figure 1: Seamless local parameterization of disk-like submeshes of the torus. The dark blue regions in the plane are identical up to translation.

these cycles *homologically non-trivial cycles*, or *non-trivial cycles* for short. Cycles which bound some region of the underlying surface will be called *trivial cycles*. An example of a trivial cycle and the two non-trivial cycles of a MCB of the KNNG of a point cloud is shown in Figure 2. At least two of the basis cycles must be non-trivial and closedness equations should *not* be formed for them. If, ideally, *exactly* two basis cycles are non-trivial, these form a basis for the first homology group of the underlying genus-1 manifold. In this case we will have $E-V-1$ independent closedness equations. When adding also the $V-1$ independent co-closedness equations for the vertices, there are in total $E-2$ linearly independent equations. These equations may be expressed as a full rank matrix A of size $E\times(E-2)$, hence they have a two-dimensional solution space in R^E, which may be computed as the nullspace of A.

There is a standard procedure for computing a cycle basis of a graph in $O(E\log E)$ time, using a spanning tree of the graph [3]. However the cycles in this basis may be quite long. This is bad for two reasons: 1) Long cycles cause the matrix A to be less sparse. 2) Long

cycles tend to be non-trivial cycles, which we do not want to force to be closed, and different (linear) equations must be formed for them, taking this into account. Luckily, there exist algorithms for computing a so-called *minimal cycle basis* (MCB), which form a cycle basis which minimizes the sum of the lengths of the cycles [22]. A MCB tends to contain very small cycles (mostly of length 3), and only two long cycles which are usually non-trivial. Computing a MCB requires $O(E^3)$ time.

The advantage in using the MCB is that if the sample is dense enough, the two longest cycles are guaranteed to be non-trivial, and the rest trivial. To formalize this, we need the following definitions, which have been used before [1] in this context:

Definition: The *medial axis* of a 2-manifold $M \subseteq \mathrm{R}^3$, denoted by $MA(M)$, is the set of points in R^3 equidistant from at least two distinct points of M.

The function $d(A,B)$ denotes the Euclidean distance between two sets A and B, i.e. the shortest distance between two points in each set.

Definition: The *local feature size* of a point p on a 2-manifold $M \subseteq \mathrm{R}^3$, denoted by $fs(p)$, is $d(p,MA(M))$.

Definition: A set of points S on a 2-manifold $M \subseteq \mathrm{R}^3$ is said to be an (ε,δ)-*sample* of M if for every $p \in M$, $d(p,S) \le \varepsilon \cdot fs(p)$, and for all $s_1,s_2 \in S$, $d(s_1,s_2) \ge \delta \cdot fs(s_1)$.

The last definition requires the sample set to be dense with respect to the local feature size, but at the same time, the samples cannot be too close to each other.

The following theorem, proved elsewhere [18], provides the guarantee we need:

Theorem [18]: Let $M \subseteq \mathrm{R}^3$ be a smooth closed 2-manifold surface. Given an (ε,δ)-sample of M, there exists a large enough value k (which is a function of ε and δ) such that the MCB of the k-nearest neighbor graph of the sample set contains exactly two cycles which are non-trivial and whose length is $\ge 4(k+3)$. All the other cycles are trivial and have length $\le 2(k+3)$. ◊

To summarize, the system of linear equations that must be solved for the one-form on the KNNG is:

$$\forall v \in \{1,..,V\} \quad 0 = \sum_{e \in \delta v} w_e \Delta x_e \tag{4}$$

and

$$\forall c \in \{1,..,C-2\} \quad 0 = \sum_{e \in \delta c} \Delta x_e \tag{5}$$

where C is the size of the MCB, and the cycles are ordered such that the two non-trivial cycles are last.

It is interesting to note that in the special case where the point cloud graph happens to be the connectivity graph of a closed manifold of F triangular faces, the MCB is just the set of all F-1=E-V-1 face boundaries but one, plus two non-trivial loops, for a total of E-V+1 cycles, as expected.

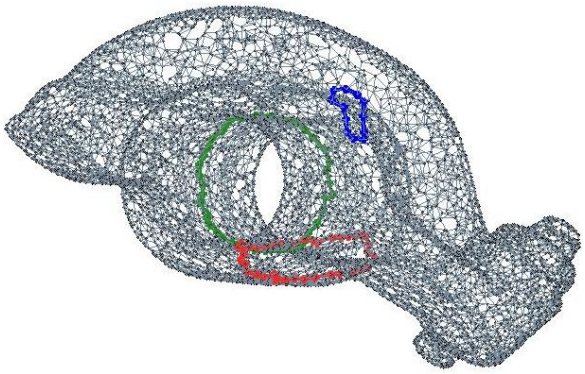

Figure 2: Three MCB cycles on a KNNG of a point cloud: trivial (blue) and non-trivial (red and green). The first should be closed and the latter two not.

4.3 Solving the harmonic one-form equations

Once the sparse coefficient matrix A representing equations (4) and (5) for the one-form have been constructed, the harmonic one-forms may be generated by computing an orthogonal basis (using a standard inner product) of the two-dimensional nullspace of A. This may be achieved by running a nullspace computation procedure, e.g. by inverse iteration on A, which may be done efficiently using a sparse LU factorization [19].

There is, however, an alternative to the nullspace computation method. Since the non-trivial cycles are also known, the basis may also be computed by solving two non-homogeneous linear systems of size E+V-1, where the right-hand side vector of the first is all zeros and a one in the row corresponding to the first non-trivial cycle, and the rhs vector of the second has all zeros, except a one in the row corresponding to the second non-trivial cycle.

4.4 Parameterizing subgraphs using the one-forms

Having computed the two independent (and orthogonal) harmonic one-forms for the edges of the KNNG G, any connected subset of vertices (which represent points in the 3D cloud) may now be parameterized. This is done by assigning any arbitrary vertex as the

origin with coordinates (0,0), and then integrating each of the two one-forms along the edges of G using any traversal strategy (e.g. DFS, BFS) to form 2D coordinates for every other vertex. Thus every other vertex in the subset is ultimately assigned 2D coordinates. Thanks to the closedness of all cycles, these are independent of the exact integration paths used.

A feature that emerges from the parameterization-by-integration method, is that if G_1 and G_2 are two connected subgraphs of G, $V_3 = V(G_1) \cap V(G_2)$ and each is parameterized using one-form integration, their parameterizations will coincide on V_3 up to a translation of the plane. This is useful, because if we independently triangulate each of the resulting 2D parameterizations using a triangulation method which is translation-invariant, the triangles will coincide on V_3. This is illustrated in Figure 3.

4.5 Meshing the point cloud

Following Floater and Reimers [15] and Zwicker and Gotsman [31], we mesh the point cloud by triangulating its 2D parameterized equivalent, and adopt that triangle structure for the 3D points. However, since our parameterization, as derived from the one-forms, is essentially doubly periodic (around the surface "handles"), we cannot easily parameterize the entire point cloud in one sweep. It is best done in (overlapping) pieces. The one-form theory guarantees that all pieces fit together seamlessly and consistently.

In practice, we mesh the point cloud in overlapping patches by choosing a number of "seed" vertices from the cloud. For each seed we extract the subgraph of all vertices connected to the seed vertex, whose distance (in the KNNG) from the seed is less than some parameter d. This subgraph is parameterized to the plane and then triangulated using 2D Delaunay triangulation. The subset of Delaunay triangles within radius $r<d$ of the seed are adopted for the 3D point cloud. The reason not to adopt *all* the Delaunay triangles is because the points along the convex hull of the patch are triangulated without knowledge of the points outside the patch, and this may lead to triangles along the patch boundary which will be inconsistent with those in neighboring patches. The entire point cloud is ultimately covered with these subsets of triangles, so care must be taken to ensure that all points are covered. This is easily achieved by taking a sufficiently large number of seeds (say $V/10$) and appropriate d and r (say d=30 and r=15). There is a fair chance that some points may participate in triangles originating in more than one patch. However, since our method guarantees that all parameterizations are identical up to a translation, these triangles will be identical, and duplicates may be simply removed at the end. Figure 3 shows the parameterization of two adjacent patches in a KNNG of a point cloud.

A pseudo-code summary of all the steps of our meshing algorithm appears in Figure 4.

5. Output Quality

It would be very useful to guarantee certain natural properties of the output mesh. First and foremost, we would like a topological property, that the output is indeed a closed genus 1 manifold. Beyond that we would like a geometric property, that the piecewise-linear mesh surface is close to the sampled surface.

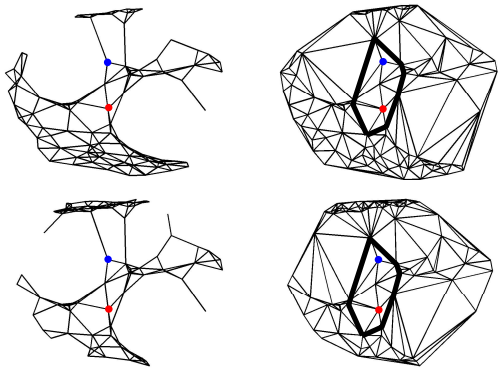

Figure 3: Triangulation of overlapping patches. **(Top left)** KNNG connectivity on parameterized vertices in patch of radius 6 around a red vertex. **(Bottom left)** Same for blue vertex. **(Right)** Delaunay triangulations of the same vertex set as on left. Note the equivalence up to translation in the marked subpatch around the colored vertices. Beyond this edge effects might creep in.

The topological guarantee is provided by the Theorem of Section 4.2. It seems that in practice the condition of the Theorem is easily satisfied, and we are able to obtain the desired closed manifold of genus 1 at relatively low sampling densities.

At this stage, we are not able to prove that the output of our algorithm, given that the above conditions on the sampling density are satisfied, is *geometrically* close to the sampled surface. However, the following intuitive argument seems to indicate that this should be the case. The parameterization of the point cloud to 2D is one which minimizes the sum of squares of the (weighted) KNNG edge lengths. This means that points which are close to each other in 3D will be close to each other in 2D. Now the subsequent Delaunay triangulation of the

2D parameterization also prefers short edges (although it does not strictly minimize this) and large angles. Thus the triangles that are formed in 2D, and subsequently in 3D, are quite small and fat. Interpolating the points using this type of triangles is more likely to position the triangles close to the surface than other types of triangles.

Input: Set of n points in 3D
Output: Closed manifold triangle mesh with genus 1 whose vertices are the input points.
Algorithm:
1. Generate G – the KNNG on the input for suitable k.
2. Compute B – the MCB of G.
3. Identify C_1 and C_2 – the two non-trivial cycles in B as the two longest cycles in B.
4. Remove C_1 and C_2 from B and populate A – the sparse coefficient matrix of the harmonic one-form equations – derived from the trivial cycles.
5. Compute Δx and Δy – an orthonormal basis for null(A).
6. $M := \varnothing$
7. for $i := 1$ to number of well distributed "seed" vertices
8. Parameterize a patch P of breadth d around seed vertex v_i in G by integrating Δx and Δy while running BFS
9. Compute D – the Delaunay triangulation of P.
10. $M := M \cup${triangles of D within radius r of v_i}.
11. end
12. Output the unique triangles in M.

Figure 4: Pseudo-code of meshing algorithm.

6. Experimental Results

We have fully implemented the meshing method described here in MATLAB (except for the MCB algorithm, which was implemented in C++ and obtained in executable form) and applied it to a number of point clouds sampled from genus 1 surface models. This was run on a 2.99 GHz Pentium 4 PC with 2GB RAM.

6.1 Some examples

Figures 5-8 show the results of our meshing algorithm applied to some point clouds sampled from some well-known genus 1 models. For example, Figure 5 shows a result on the "rocker arm" model, which was decimated down to 7,634 points. This point cloud was meshed in 4.5 minutes. It is a perfect manifold, containing 15,268 triangles. However, it is not watertight, as the surface may self-intersect. This can be resolved by independent post-processing. We applied the methods of Surazhsky and Gotsman [28] and Surazhsky et al. [29], originally designed for a remeshing scenario. These improve the regularity of the connectivity by edge flipping, and the regularity of the geometry by "sliding" points along the imaginary manifold surface. Some of these results are shown in the same figure, as in Figure 7. All of the resulting meshes in Figures 5-8 are closed manifolds of genus 1.

6.2 Other degrees of freedom

There are a few degrees of freedom in our algorithm that may be exploited. The first is the choice of weights used in the co-closedness equations (4). The geometry of the input point set is taken into account via these weights. We used $w_e = 1/l_e$, where l_e is the Euclidean length of edge e. This tends to position the points in the plane such that the 2D edge lengths are close to the 3D edge lengths. It is also possible to use other decreasing functions, such as $w_{ij} = \exp(-l_e)$. We did not observe a major difference.

Another degree of freedom is in the choice of the two independent one-forms from the two-dimensional space of harmonic one-forms. The nullspace routine that we employ computes two vectors which are orthonormal in R^E. Theoretically, we could have used any other two linearly independent vectors from this space, namely the result of an arbitrary non-degenerate affine combination of the two orthonormal vectors. Since we integrate these vectors to produce x and y coordinates, and then use Delaunay triangulation to produce the mesh triangles, this could affect the result, as Delaunay triangulation is not affine-invariant.

6.3 Performance

The most computationally-challenging components of our meshing algorithm are the computation of a KNNG on V points in R^3, the computation of a minimal cycle basis (MCB) on E=O(kV) edges and computation of the nullspace of a sparse matrix of size ExE. Computing the KNNG may be done in O(Vlog$V$) time using standard spatial data structures [7,27]. Computing the MCB theoretically requires $O(E^3)$ time, but the implementation that we received from Michail, based on [24], runs in a matter of minutes for graphs of up to 15,000 vertices. Computing the nullspace of the resulting sparse matrix using the methods described in [19] required just a few seconds. Reconstructing the manifold required a minute or so, and the post-processing improvements a few seconds. In total, the entire algorithm runs in minutes on clouds of 15,000 points, and we believe it may be further optimized by an order of magnitude.

6.4 Implementation details

There are a number of important parameters in our algorithm which have to be chosen carefully. The first is k – the degree of the vertices in the KNNG constructed on the sample points. The first concern is that k should be sufficiently large for the graph to be 3-connected, otherwise we may get degeneracies in the parameterization. On the other hand, in order that the harmonic equations give the correct result, there should be only two identifiable non-trivial cycles in the MCB. Anything else will affect the dimension of the nullspace of A, and non-harmonic one-forms will result. This means that the sampling density should be sufficiently high and k sufficiently small such that any non-trivial cycle is sufficiently long. As demonstrated by Gotsman et al. [18], the probability the MCB containing more than two non-trivial cycles is almost nil, for most reasonable values of k. A good value of k will generate a MCB of *short* (length at most $2(k+3)$) trivial cycles and *long* (length at least $4(k+3)$) non-trivial cycles. In practice, we found that $k = 7$ or 8 was a good choice, since we adopt the edge (i,j) only if both $i \in N(j)$ and $j \in N(i)$.

Since we use the Delaunay triangulation on the resulting 2D parameterization of the point set, we must be careful of the degeneracies which might arise in these types of parameterizations, namely co-linearities and co-circularities. The former can cause triangles to "disappear" and the latter can cause the Delaunay triangulation to be non-unique, possibly inconsistent between triangulations of overlapping patches. We have found that simulation of simplicity - adding a small amount of noise to the 2D parameterization - is a good remedy.

7. Discussion and Conclusion

We have presented a method to mesh a point cloud sampled from a closed manifold of genus 1. Under very mild conditions on the sampling density, the algorithm will generate a closed manifold of genus 1. We believe that under the same conditions, it is possible to prove a bound on the geometric approximation quality of the manifold relative to the underlying surface.

Seemingly the main drawback of this algorithm is that it is *global*, namely it solves simultaneously for the solution on the *entire* mesh. This requires manipulation and computation of very large structures, and may potentially not scale well. However, the globality is also a distinct advantage. It guarantees that the result is a perfect manifold, containing no artifacts that might arise from seams, were the point cloud partitioned and meshed in pieces. In practice, meshing point clouds of up to 20,000 points did not require more than a few minutes on a state-of-the-art PC. Our sampling density condition is also a global condition, and this is possibly why it is so mild.

The most intriguing question is how to generalize our method to surfaces of genus $g>1$. It is possible to assemble the same set of equations as described here to generate the space of harmonic one-forms, which may be integrated to form a 2D parameterization. However, this space has dimension $2g$, and will contain $2g$-2 singularities [20,16], which will lead to artifacts in the parameterization. This is the subject of current investigation.

References

[1] N. Amenta, M. Bern, and M. Kamvysselis. *A new Voronoi-based surface reconstruction algorithm.* Proceedings of SIGGRAPH 1998, pp. 415-421, 1998.

[2] N. Amenta, S. Choi and R. Kolluri. *The power crust*, Proceedings of 6th ACM Symposium on Solid Modeling, pp. 249-260, 2001.

[3] B. Bollobas. *Modern graph theor*y. Graduate Texts in Mathematics, Springer Verlag, 1998.

[4] J.-D. Boissonnat. *Geometric structures for three-dimensional shape representation.* ACM Transactions on Graphics, 3(4):266-286, 1984.

[5] J.-D. Boissonnat and F. Cazals. *Smooth surface reconstruction via natural neighbour interpolation of distance functions.* Proc. 16th Annual ACM Sympos. Comput. Geom., pp. 223-232, 2000.

[6] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans. *Reconstruction and representation of 3D objects with radial basis functions.* Proceedings of SIGGRAPH, pp. 67-76, 2001.

[7] K. Clarkson. *Fast algorithms for the all nearest neighbors problem.* Proceedings of 24th IEEE FOCS, 226-232, 1983.

[8] B. Curless and M. Levoy. *A volumetric method for building complex models from range images.* Proceedings of ACM SIGGRAPH 1996, pp. 303-312, 1996.

[9] T. K. Dey and S. Goswami. *Tight cocone: A water-tight surface reconstructor*. Journal of Com-

puting and Information Science in Engineering, 3:302-307, 2003.

[10] T. K. Dey. *Curve and surface reconstruction.* In J. E. Goodman and J. O'Rourke, editors, Handbook of Discrete and Computational Geometry. CRC Press, 2004.

[11] A. N. Hirani. *Discrete Exterior Calculus.* Ph.D. Thesis, California Institute of Technology, 2003.

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. *Surface reconstruction from unorganized points.* Proceedings of SIGGRAPH 1992, pages 71-78, 1992.

[13] M.S. Floater. *Parameterization and smooth approximation of surface triangulations.* Computer Aided Geometric Design, 14:231–250, 1997.

[14] M. S. Floater and M. Reimers. *Meshless parameterization and surface reconstruction*, Computer Aided Geometric Design 18:77-92, 2001.

[15] M.S. Floater and K. Hormann. *Surface Parameterization: a Tutorial and Survey.* In Advances in Multiresolution for Geometric Modelling, N. A. Dodgson, M. S. Floater, and M. A. Sabin (eds.), Springer-Verlag, pp. 157-186, 2004.

[16] S.J. Gortler, C. Gotsman and D. Thurston, D. *Discrete one-forms on meshes and applications to 3D mesh parameterization.* Computer Aided Geometric Design, 23(2):83-112, 2006.

[17] C. Gotsman, X. Gu and A. Sheffer. *Fundamentals of spherical parameterization for 3D meshes.* ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003), 2003.

[18] C. Gotsman, K. Kaligosi, K.Melhorn, D. Michail and E. Pyrga. *On the minimal cycle basis of sampled surface manif*olds. Preprint, 2005.

[19] C. Gotsman and S. Toledo. *On the computation of the nullspace of a sparse non-normal matrix.* To appear in SIAM J. Matrix Analysis and Applications, 2006.

[20] X. Gu and S.-T. Yau. *Computing conformal structures of surfaces.* Communications in Information and Systems 2(2):121-146, 2002.

[21] K. Hormann and M. Reimers. *Triangulating point clouds with spherical topology*. Proceedings of Curve and Surface Design, Saint Malo, 2002.

[22] T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. *A faster algorithm for minimum cycle basis of graphs.* Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP), 2004.

[23] R. Kolluri, J.R. Shewchuk and J.F. O'Brien, *Spectral surface reconstruction from noisy point clouds*. Proc. Symposium on Geometry Processing, pp. 11-21, 2004.

[24] K. Mehlhorn and D. Michail. *Implementing minimum cycle basis algorithms*. Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA), 2005.

[25] C. Mercat. *Discrete Riemann surfaces*. Communications of Mathematical Physics, 218(1):77-216, 2001.

[26] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. *Multi-level partition of unity implicits.* ACM Transactions on Graphics, 22(3):463-470, 2003. Proceedings of SIGGRAPH.

[27] S. Skiena. *The algorithm design manual.* Telos, 1997.

[28] V. Surazhsky and C. Gotsman. *Explicit surface remeshing.* Proceedings of the ACM/Eurographics Symposium on Mesh Processing, 2003.

[29] V. Surazhsky, P. Alliez and C. Gotsman. *Isotropic remeshing of surfaces: A local parametrization approach.* Proceedings of the International Meshing Roundtable, 2003.

[30] W.T. Tutte. *How to draw a graph.* Proceedings of the London Mathematical Society, 13(3):743-768, 1963.

[31] M. Zwicker and C. Gotsman. *Meshing point clouds using spherical parameterization.* Proceedings of the Eurographics Symposium on Point-Based Graphics, 2004.

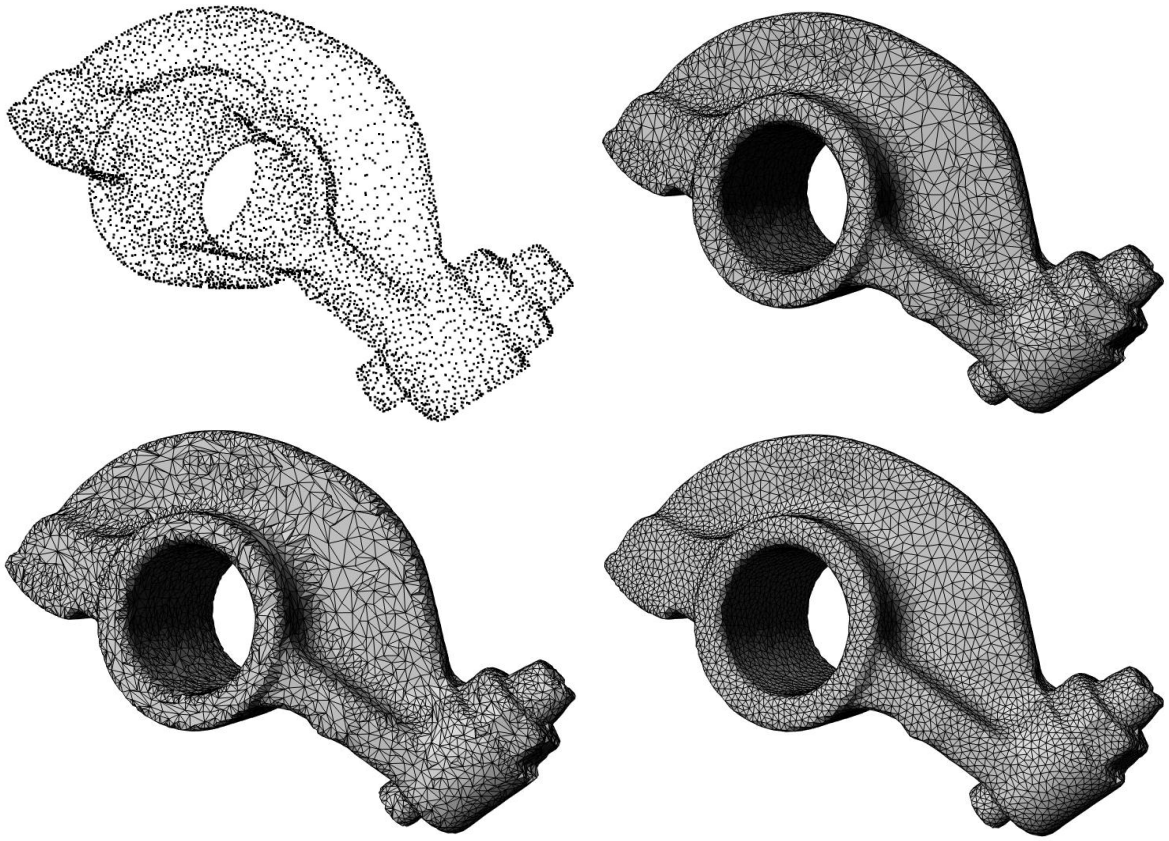

Figure 5: Meshing of the 7,634 point "rocker arm" model. **Top left:** Input point cloud. **Bottom left:** Mesh output of our algorithm using k=7, t=10, d=15 and no simulation of simplicity (no addition of noise). **Top right:** Improvement by edge flips only (points not moved). **Bottom Right:** Improvement by edge flips and point sliding. Latter two using algorithm of Surazhsky et al [29].

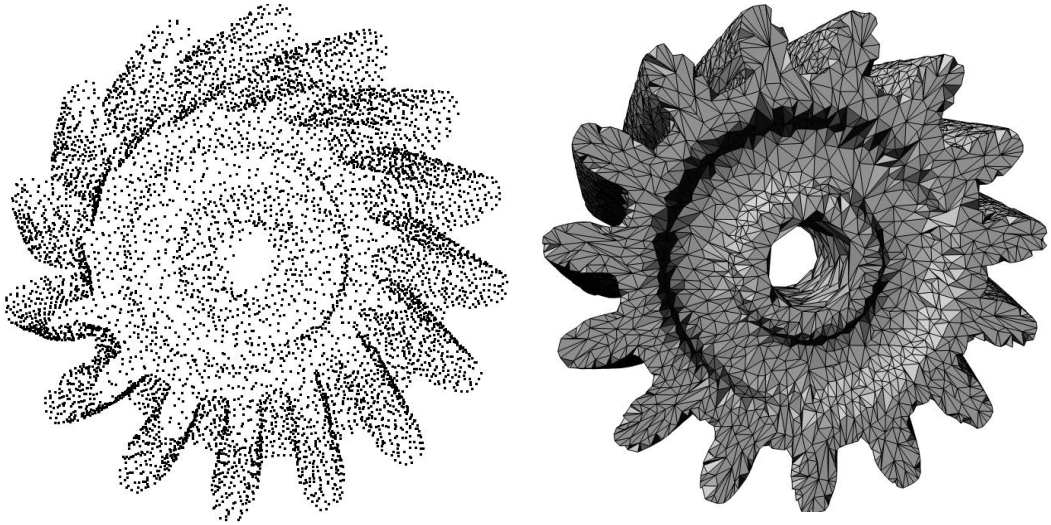

Figure 6: Reconstruction of the 5,044 point "gear" model. Parameters used were k=7, t=10, d=10 and no simulation of simplicity.

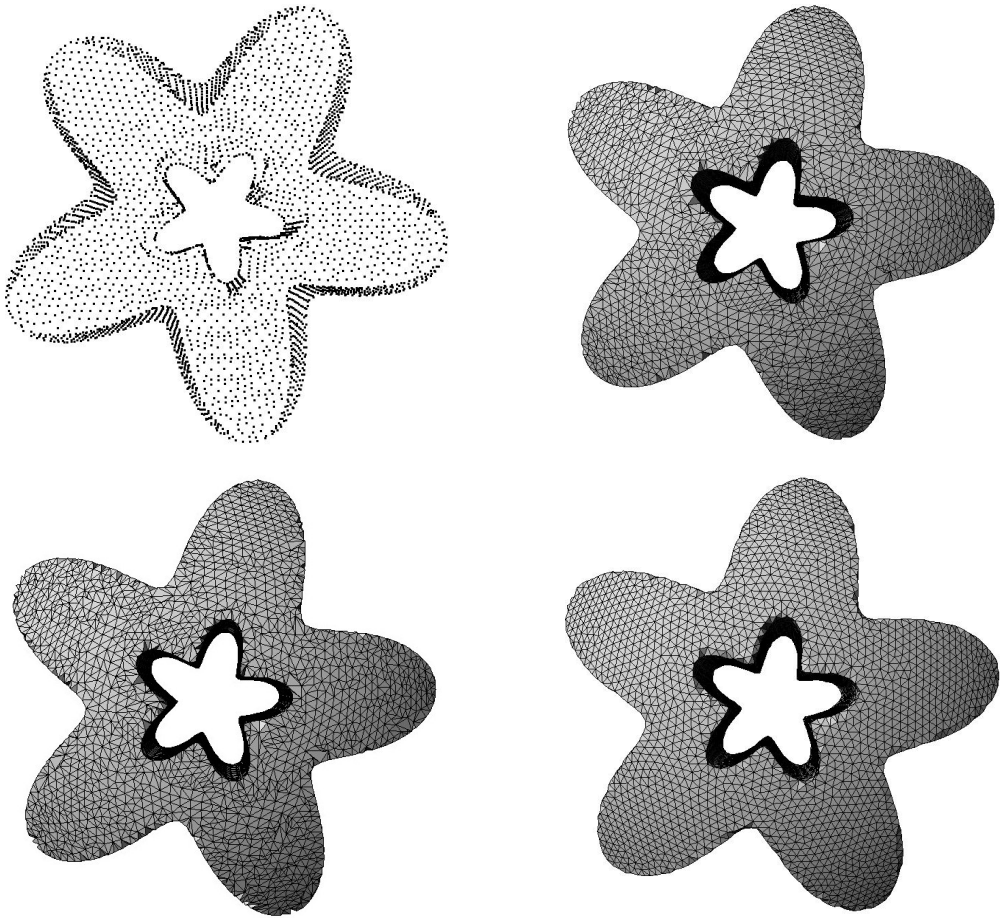

Figure 7: Same as Figure 5 for the 5,982 point "trim star" model. Parameters used were k=7, t=10, d=10 and random noise with variance 0.001 for simulation of simplicity.

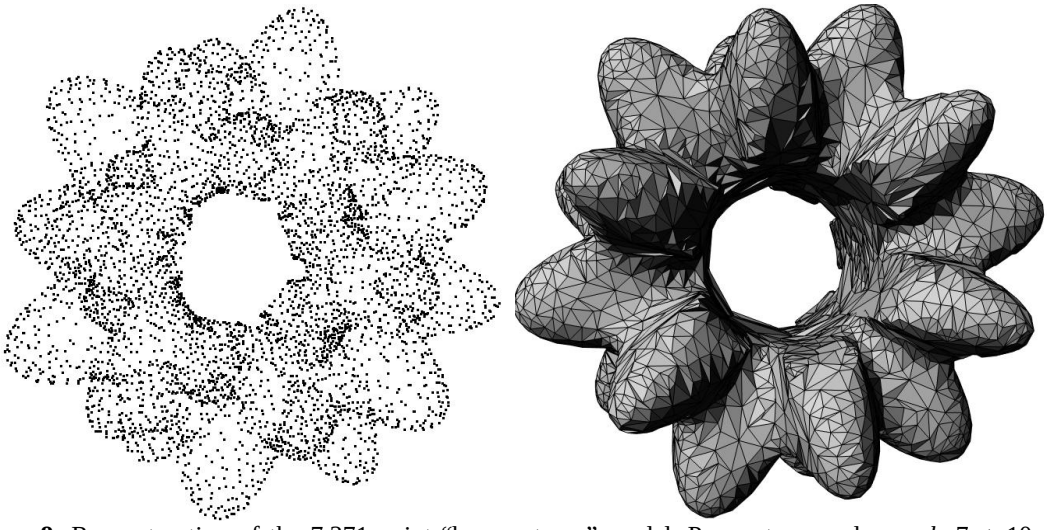

Figure 8: Reconstruction of the 7,371 point "bumpy torus" model. Parameters used were k=7, t=10, d=10 and no simulation of simplicity.